

Lab Assignment & Solution



Cybersecurity Professional Program
Introduction to Python
for Security

File System & Error Handling

PY-04-LS2

Error Handling

Note: Solutions for the instructor are shown inside the green box.



Lab Objective

Understand how error detection and handling control code execution.



Lab Mission

Practice handling error conditions that may occur in Python code.



Lab Duration

15–20 minutes



Requirements

- Working knowledge of basic programming
- Working knowledge of exception handling



Resources

- Environment & Tools
 - Windows
 - PyCharm
 - Python 3



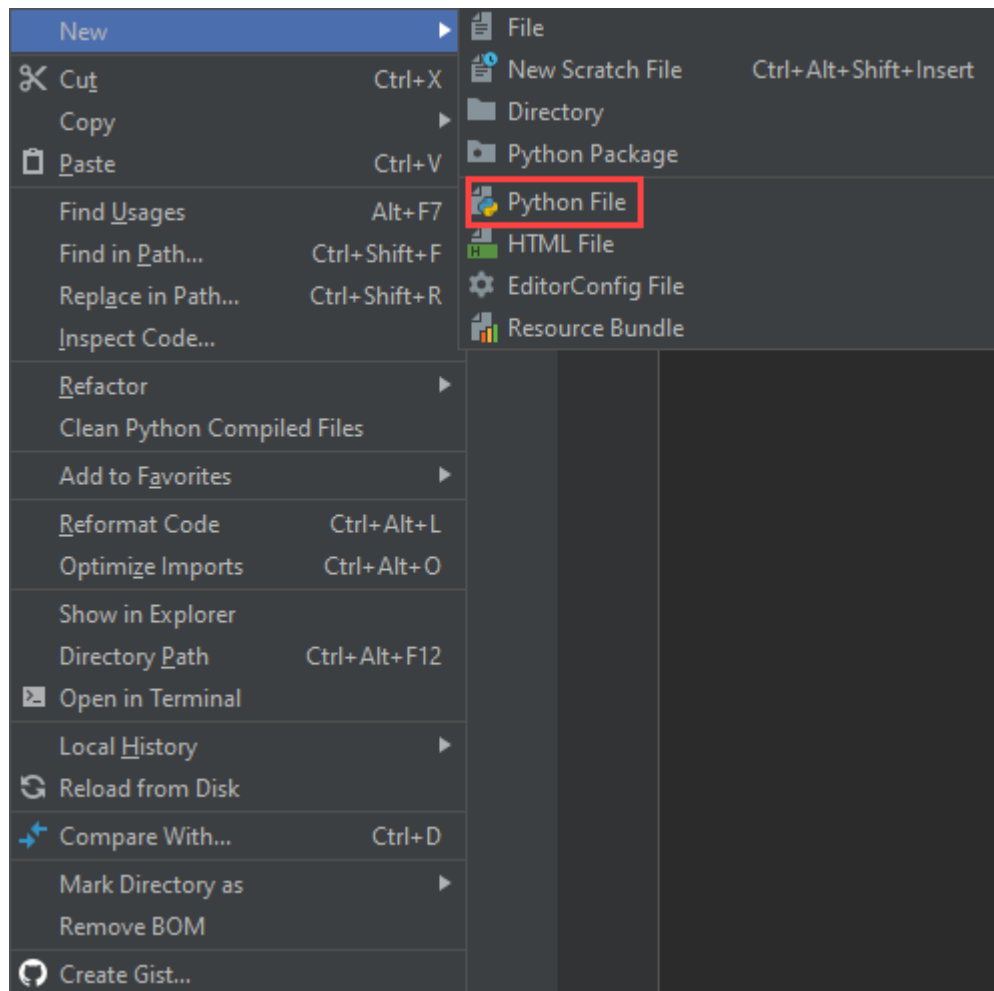
Textbook References

- Chapter 4: File System and Error Handling
 - Section 1: Error Handling

Lab Task: Product Calculation in Python

Write a program that calculates the product of four numbers provided by the user and prints the result. Use **try** and **except** statements to ensure that the program will not fail when the input is not a valid number.

- 1 Create a new Python file in PyCharm by right-clicking the project you created and selecting **New > Python File**.



- 2 Declare the variable **product** and assign it an integer value of 1.

```
product = 1
```

- 3 Create a **for** loop that performs four iterations.

```
product = 1
for i in range(4):
```

- 4 In the **for** loop, ask the user to provide a number, cast that number to an integer, and assign it to a new variable. Multiply each input by the **product** variable and assign the result to the same variable.

```
product = 1
for i in range(4):
    num = int(input("Enter a number: "))
    product *= num
```

- 5 Place the user input and mathematical operation in a **try** block. Make sure that you use indentation when placing it in the **try** block.

```
product = 1
for i in range(4):
    try:
        num = int(input("Enter a number: "))
        product *= num
```

- 6 Create an **except** block that prints a message to the console when user input is a non-integer.

```
product = 1
for i in range(4):
    try:
        num = int(input("Enter a number: "))
        product *= num
    except :
        print("The input is not a valid number")
```

- 7 Run the code from Step 6, input an integer, then input a non-integer, and observe the results.

The last input, as shown below, is a non-integer. The error is caught by the **except** statement in the **try** block and triggers the invalidation message to appear.

```
Enter a number: 4
Enter a number: 3
Enter a number: 2
Enter a number: A
The input is not a valid character
```

- 8 Add a line to print the four numbers' product and cast the **integer** to a string.
- 9 Rerun the code, but this time input integers only. Observe the results.

```
product = 1
for i in range(4):
    try:
        num = int(input("Enter a number: "))
        product *= num
    except:
        print("The input is not a valid number")
print("The product of the 4 numbers is: " + str(product))
=====
Enter a number: 4
Enter a number: 3
Enter a number: 2
Enter a number: 1
The product of the 4 numbers is: 24
```